CS5721 SOFTWARE DESIGN

TEAM-BASED PROJECT

Specification V1

Semester 1: 2021-2022

J.J. Collins

24th September 2021 (Week 3)

1. Objectives

1. To apply an Object-Oriented (OO) Analysis and Design (OOAD) method using the Unified Modelling Language (UML) to the development of an enterprise software system.
2. To apply architectural and design patterns where appropriate.
3. To identify the strengths and liabilities of the OOAD approach using the UML through study, practice, and reflection.
4. Gain insights into team dynamics and project management from the experience.

2. Scenario

You have been commissioned by a client to either develop a new system, or adapt an existing system to incorporate changes in the business model. A list of examples is as follows:

- Simulations.
- Smart city / smart health / smart home / smart ….
- Recommender system for retail
- A software tool such as a diagram compiler,

The system should have sufficient "business rules" to make the application compute intense, as opposed to being one that primarily post queries and updates to a database from a GUI. One example of a set of business rules would be differing discount policies for various customer categories in a retail system.

3. Specification

You are required to submit a specification and subsequent implementation, which contains:

i. Front cover with business scenario title, student names and IDs, and module details.
ii. Table of contents.

1. A one-page (maximum) narrative description of your business scenario.
2. A one-page (maximum) discussion on the software lifecycle model adopted to manage the process of software development.
3. Lightweight project plan with allocation of roles clearly stated.
4. **Requirements**
    - Functional:
        - A listing of requirements and their allocation to use cases.
        - One or more use case diagram(s). Do not use decomposition.
        - For each use case, a semi-structured use case description, see the template at the end of chapter A2 in Bennett, McRobb, and Farmer (4th Edition)
        - For two key use cases, the description must be based on the template distributed in the lecture notes Week 3.
    - A listing of Non-functional requirements.
    - A short discussion on tactics that one might consider adopting to support specific quality attributes. See the chapter on tactics in Software Architecture in Practice by Bass et al. (2010).
    - A brief selection of:
        - 2 sample GUI prototypes - TOKEN ACKNOWLEDGEMENT of the role of prototyping in requirements elicitation.
5. A two page (maximum) discussion of architectural styles/patterns **used/considered** for the system architecture. You must include the Model View Controller (MVC) architectural patterns.
    - One of the patterns discussed must be independently researched i.e. not be one of those presented in lectures - MVC, Broker, and Scheduler.
6. A two-page (maximum) discussion of system architecture to include at least one high-level package diagram providing an architectural perspective. You should also discuss technology pipeline that will be used to host deployment i.e. web server, application server, Enterprise Information System (EIS) aka database, message bus, etc.
7. Lightweight **analysis sketches**: use an object-oriented methodology with UML. Include the following descriptions / diagrams where appropriate, or show why not appropriate:
    - List of candidate objects derived using Data Driven Design (DDD).
    - Sketch of the analysis class diagram, with significant methods and attributes identified. The sketch should demonstrate inheritance, aggregation and composition, associations,

dependencies, visibility, etc. There should be no duplication of attributes or methods in the diagram. The diagram should contain at least two class interfaces.

- A sketch of a communication diagram
- A sketch of a sequence diagram.
- Entity Relationship diagram with cardinality - Not covered in CS5721!
- **SECTION 7 MUST BE COMPLETED BEFORE STARTING SECTION 8-10.**

8. **A listing in tabular format of classes in each package, their authors, and lines of code. Also include total lines of code developed (a) for the application as a whole, and (b) per team member.**
9. **Code**: the implementation is a light-weight proof of concept prototype. Implement those classes necessary to support a subset of the use cases.
    - Include coding fragments in this section of the report to illustrate implementation of MVC, design patterns, and other "interesting" aspects of the implementation. Also include screen shots of the GUI/UI where one was developed.
    - The team should use a version control system such as Github, CVS etc. Include screen shots with brief description that visualise history of commits
        a. Commits should be distributed across the coding weeks and from all team members
    - Automated testing is mandatory, for example, Junit / Nunit for the implementation of automated test cases - 2 should be sufficient, TOKEN ACKNOWLEDGEMENT of the role of Test Driven Development (TDD) in Continuous Integration (CI).
    - You must keep a "diary" as illustrated in Table 1 that briefly describes the contribution of each team member for each week, issues arising, and the plan for the following week

Table 1: sample template for the diary

Week X	Contributions for this week	Issues arising this week	Plan for next week
Mem 1			
Mem 2			
Mem 3			
Mem 4			
Notes			

→ Include this diary in the report as an Appendix

- Evidence of code inspections is necessary. This could be achieved through the kreoh platform which will be introduced in Week 4.

10. **Code - Added Value**: document implementation for concepts not covered in the lectures, for example:
    - Object Relational Mapping (ORM)
    - REST architectural pattern
    - Concurrency through threading or OpenMP.
    - Security
    - Software Metrics
    - Dev Ops
    - Other

    Include coding fragments and/or screen shots with brief descriptions in this section of the report to illustrate implementation of Added Value.
11. Recovered **architecture and design blueprints** based on the implementation but can also specify concepts that could be developed in future releases. Draw blueprints of:
    a. Architectural diagram
    b. Design-time class diagram.
    c. One state chart for an object in either the sequence or communication diagram, with annotated transition strings.
12. Component and deployment diagrams as supported by the UML.
14. Critique: compare and contrast the analysis sketches in (8) versus the blueprints created in (12).
15. References.

PLEASE NOTE: implementation always follows on from design. Due to time constraints, we cannot adhere to this chronological ordering in this project because design will only be fully covered in lectures by the end of week 10. If we waited until Week 10 to finish the design phase of the project and then start coding, there would not be sufficient time for a meaningful implementation. Therefore, for this project only, implementation is guided by analysis artefacts - see section 8. We recover architecture and design post implementation.

5. Notes and Guidelines

- This assignment **constitutes approximately 50%** of the total marks awarded for this module.
- You will work in a team of 4.
- Email me with the subject "CS5721 Team" by 10am Mon 27th Sept. (Week 4) at the latest should you wish to be allocated to a team.
- **The award of an F or NG will automatically result in the same award for the module.**
- **Submission deadline is 12:00 Monday 22nd November 2021 – Teaching Week 12.**
- NO SUBMISSIONS WILL BE ACCEPTED AFTER THIS DATE!
- Submission is via the Sulis Assignment tool.
- You MAY be required to provide the lecturer with a walk through of your project submission during an interview in Teaching Week 13-15.
    - The project will be awarded an F grade if a walkthrough is not provided when requested to do so.
- Programming language and development environment is at your discretion.
- Presentation is critical. The report should be concise, easy to read, with diagrams that conform to aesthetic guidelines, i.e. minimise line crossings.
- If the presented work is not to the standard expected at this level, the submitted work will be severely penalised. Please ensure that:
    - Layout conforms to specification.
    - Page numbers in table of contents and throughout report.
    - Spell and grammar checks done.
- Analysis **sketches** MUST be drawn on paper or a whiteboard, captured using a mobile phone camera, and inserted in jpeg/gif/etc form into the report.
- You must use a UML workbench for diagrams produced in section 11 and 12 of the report. Numerous lists of community edition tools are available on the net, select one and justify your choice in the report. Using Word or PowerPoint is definitely excluded for diagramming.
- You may **REUSE** content from an existing project, but you must:
    - Demonstrate an in-depth understanding of structure and dynamics.
    - Adhere to the rules with respect to the prevention of plagiarism.
    - Clearly differentiate between what already exists and added value.
        - See "Cite it Right", available through the UL Library Catalogue.
    - Demonstrate added value – for example, by extending the feature set

- Quality not quantity. Depth not breadth. Do not attempt to develop a full enterprise system.
    - KEY ENGINEERING PRINCIPLE: Keep It Simple …. (KISS).
- Marks are awarded for having attempted to use the methodologies and tools, for developing a lightweight proof of concept prototype, and evaluating the utility of diagrams. Marks are not awarded for the complexity or expansiveness of your selected scenario.
- Separation of the business classes and UI is critical. Make sure that you have read and adopted one of the standard approaches to the design of user interfaces such as the Model View Controller (MVC) approach.
- Supporting quality attributes such as performance and extensibility is key for a successful submission. You may consider submission of additional artefacts that have not been explicitly requested such as timing diagrams. The submission of additional material that has relevance and adds value will be rewarded.
- The lecturer reserves the right to make minor amendments to the assignment up to and including week 8. Amendments will be sympathetic to any work done to date.
- Use the cloud for the report.

PLAGIARISM WILL AUTOMATICALLY RESULT IN AN F GRADE FOR THE MODULE AND REFERRAL TO THE DISCIPLINARY COMMITTEE.

Start Now

Work Clever

ALL TEAM MEMBERS MUST CONTRIBUTE EQUALLY TO THE IMPLEMENTATION.

All team members must contribute EQUALLY to the writing of the report.

5. Note on Team Dynamics

- Professional Practice requires each team member to contribute as per agreed work schedule and timelines, and to notify the team and lecturer should a slippage be likely.
- Professional practice requires a team to escalate a problem with a team member to their "line manager" who happens to be the lecturer in this instance.
    - Reported problems immediately which will be then handled in a sensitive manner.
- The primary mechanism to resolve problems with team dynamics will be to resort to individual submissions.
- If you are required to provide a project walk through, and
    - It becomes evident during the walkthrough that one of the team members did not contribute equally to the implementation,
    - The team did not notify the lecturer of a problem with team dynamics

  the following penalties will be imposed:
    - F for the non-contributing member
    - Other team members will be capped at a C grade.
- Please note that the Marking Scheme (in a separate handout) clearly states that each team member will be awarded an INDIVIDUAL grade that reflects the value of their contribution.
    - Thus, don't panic if aspects of the project appear weak to you for which you are not primarily responsible.
    - But please do lend a helping hand - support and mentor, to team members who may be struggling but are trying.

- Division of labour amongst team members is essential to a successful outcome, as is the necessity to commence immediately. Work clever.
- Collaboration between team members will be challenging and difficult. Consider using an app such as Zoho and instant messaging to help.
- All team members should attend the lab session and use the full hour as one of the scheduled weekly meetings.

6. Suggested Project Roles

Table 2

	Role	Description	Designated Team Member
1	Project Manager	Sets up group meetings, gets agreement on the project plan, and tracks progress.	
2	Documentation Manager	Responsible for sourcing relevant supporting documentation from each team member and composing it in the report.	
3	Business Analyst / Requirements Engineer	Responsible for section 6 - Requirements.	
4	Architect	Defines system architecture	
5	Systems Analysts	Creates conceptual class model	
6	Designer	Responsible for recovering design time blueprints from implementation.	
7	Technical Lead	Leads the implementation effort	
8	Programmers	Each team member to develop at least 1 package in the architecture	ALL
9	Tester	Coding of automated test cases	
10	Dev Ops	Must ensure that each team member is competent with development infrastructure, e.g. GitHub, Bamboo, etc.;	

7. Suggested High Level Project Plan

Table 3

Week	Workflow
3	Setup team roles, agree on scenario, learn to use Github, research on existing projects, etc.
4	Requirements
5	Analysis
6	High level architecture
7	Coding Iteration 1: basic infrastructure and 2 key use cases
8	Coding Iteration 2: 2 more use cases
9	Coding Iteration 3 Another use case and MVC (GUI)
10	Coding Iteration 4: Another use case and Added Value
11	Architecture and Design Recovery